REMARKS

Applicant is in receipt of the Office Action mailed February 25, 2005. Claims 43, 44, and 46-60 were rejected. Claims 43, 44, and 57-60 have been amended. New claims 61-64 have been added. Claims 43, 44, and 46-64 are currently pending in the application. Reconsideration of the case is earnestly requested in light of the following remarks.

Claim Objections

Claim 43 was objected to because of a grammatical error. Applicant has amended claim 43 to correct the error.

Section 102 Rejections

Claims 43 and 57-60 were rejected under 35 U.S.C. 102(e) as being anticipated by U.S. Patent No. 6,247,013 to Morimoto. Applicant respectfully traverses this rejection.

As per claim 43, the claim recites in part, "receiving user input specifying the data source, wherein the user input is received to a program development environment during creation of an executable program". Thus, the user input specifying the data source is clearly recited in the claim as user input that is received to a program development environment during the creation of a program. For example, the Summary of the Invention in the present application describes that, "In one embodiment, a developer of a program may specify a data source and/or data target during development of the program" (first paragraph). "In various embodiments the data source/target information may be provided in any of various ways. For example, many programming environments include a user interface editor or window for designing a graphical user interface. The developer may interact with the user interface editor window to specify the data source/target" (fifth paragraph).

Morimoto does not teach receiving user input specifying a data source, where the user input is received to a program development environment during creation of an executable program. In fact, Morimoto's invention appears to be totally unrelated to the creation or development of a program. Morimoto teaches that a web browser program may be used to access and display a web page. Such a web page may comprise an anchor

which is associated with a URL denoting the location of a document on the Internet. The user can click on the anchor, e.g., using a mouse button, to cause the web browser program to download the document and display it (Col. 1, lines 5-30).

The current Office Action attempts to equate the act of the user clicking on the anchor with receiving user input specifying a data source. However, Applicant notes that the user input in Morimoto is received to the web browser program itself, not to a program development environment, and is received while the web browser program is running. In contrast, claim 43 recites that user input is received during creation of a program, not while the program is running, and is received to a program development environment (which is used to create the program), not to the program itself.

The Office Action states that, "It is understood that it is arbitrary as to the circumstances by which the user specifies the data source, and that in fact, the user may specify such a data source via a program development environment during creation, and specifically debugging, of a program such as the helper or plug-in application (for example, see column 1, line 60 – column 2 line 34 of U.S. Patent No. 5,903,728 to Semenzato)." Applicant respectfully disagrees. The helper or plug-in application must first be created before it can be executed in the context of a debugger as described in Semenzato (see Col. 2, lines 18-21). User input received while the helper or plug-in application is being executed, whether in the context of a debugger or otherwise, does not constitute user input received during creation of the helper or plug-in application. Furthermore, even if a program development environment were used to debug a helper or plug-in application for a web browser program, the user input specifying the URL, i.e., the user input clicking on the anchor, would still be received by the web browser program. The user input specifying the URL would not be received to the helper or plug-in application being debugged, and even if it were received to the helper or plug-in application, it would not be received via the program development environment as asserted in the Office Action.

Applicant thus submits that Morimoto does not teach the element of, "receiving user input specifying the data source, wherein the user input is received to a program development environment during creation of an executable program," as recited in claim 43.

Amended claim 43 further recites the elements of, "programmatically selecting a GUI element for inclusion in the program after receiving the user input" and "displaying the selected GUI element in the program." These claim elements are also not taught by Morimoto. The Office Action states that,

> "Morimoto discloses that upon specifying the anchor, and consequently its associated URL, is determined whether or not the browser can process and display the type of data within the file, and if not, the browser selects and instantiates a specific application, such as a plug-in or helper application, based on the type of data. It is understood that the plug-in or helper application may comprise its own GUI elements in order to display the data...Consequently, Morimoto is considered to teach to one of ordinary skill in the art a method for...programmatically selecting a plug-in or helper application, and consequently its GUI elements, after receiving the user input..."

Applicant agrees that Morimoto teaches the browser selecting and instantiating a specific plug-in or helper application based on the type of data. Applicant also agrees that the plug-in or helper application may comprise its own GUI elements in order to display the data. However, Applicant disagrees that this amounts to "programmatically selecting a GUI element" as recited in amended claim 43. No programmatic selection of a particular GUI element takes place in Morimoto. Instead, the browser selects an application (not a GUI element) and instantiates the application, thus causing the GUI elements in the application to be displayed. However, the browser has no control over which particular GUI elements are displayed in the application, i.e., does not select any particular GUI element as recited in claim 43.

Furthermore, amended claim 43 recites that the GUI element is programmatically selected for inclusion in the program. Claim 43 also recites that the GUI element is programmatically selected after receiving the user input. In contrast, a plug-in or helper application as taught in Morimoto already includes its own GUI elements even before any user input is received to specify a URL. Morimoto contains no teaching or suggestion whatsoever that the GUI elements in a plug-in or helper application are programmatically selected after receiving the user input specifying the URL. Morimoto also contains no teaching or suggestion whatsoever of programmatically selecting a GUI element for inclusion in the plug-in or helper application. Instead, the plug-in or helper application is simply activated and the GUI elements which were already included in the

plug-in or helper application before the user input was received are displayed. Therefore, Morimoto does not teach, "programmatically selecting a GUI element for inclusion in the program after receiving the user input".

Applicant thus respectfully submits that Morimoto does not teach all the elements of claim 43, and that claim 43 is therefore allowable over Morimoto, for at least the reasons discussed above. Inasmuch as claim 57 recites similar elements as claim 43 (e.g., "programmatically determining a graphical user interface (GUI) element [not an application] operable to display data from the specified data source for inclusion in the program"), and claim 59 also recites similar limitations as claim 43, Applicant submits that claims 57 and 59 are also allowable over Morimoto, for at least the reasons discussed above.

Claim 58 recites similar limitations of receiving user input specifying a data source during development of a program and programmatically determining a GUI element operable to display data from the specified data source, but where the program is a graphical program, "wherein the graphical program comprises a block diagram and a user interface panel, wherein the block diagram comprises a plurality of connected nodes which visually indicate functionality of the graphical program".

The Office Action asserts that, "the plug-in or helper application is considered a graphical program comprising a plurality of interconnected nodes that visually represent functionality of the plug-in. For example, the GUI of the plug-in is understood to comprise a plurality of interconnected nodes, which generate a display of the GUI elements, and which thus visually represent functionality of the graphical program." However, Applicant submits that Morimoto contains no teaching or suggestion whatsoever that the plug-in or helper application is a graphical program that comprises a block diagram and a user interface panel. In fact, the concepts of graphical programming, graphical programs, and block diagrams are completely absent from Morimoto. As per the assertion that "the GUI of the plug-in is understood to comprise a plurality of interconnected nodes," Applicant notes that claim 58 actually recites that the nodes are comprised in a block diagram of the graphical program, not a GUI or user interface panel. A block diagram is not at all the same thing as a GUI and is not taught in Morimoto.

Applicant thus respectfully submits that Morimoto does not teach all the elements of claim 58, and that claim 58 is therefore allowable over Morimoto, for at least the reasons discussed above.

As per claim 60, the claim recites in part, "receiving user input specifying a data source and data target, wherein the data source and data target are the same". The Office Action asserts that, "Additionally it is understood that the user may implement the plug-in or helper application to publish data to the data target, as known in the art." Applicant respectfully disagrees. Morimoto contains no teaching or suggestion of receiving user input specifying a data source and data target, where the data source and data target are the same. Morimoto also contains no teaching or suggestion of using the plug-in or helper application to publish data to a data target. Applicant is also unaware of other art that teaches the use of a plug-in or helper application such as described in Morimoto to publish data to a data target.

Furthermore, Morimoto does not teach, and Applicant is unaware of other art that teaches, configuring a GUI element to publish data to a data target, as recited in claim 60 ("programmatically configuring the GUI element to receive and display data from the specified data source and publish data to the specified data target").

Applicant thus respectfully submits that Morimoto does not teach all the elements of claim 60, and that claim 60 is therefore allowable over Morimoto, for at least the reasons discussed above.

Claims 43-44, 46-58, and 60 were rejected under 35 U.S.C. 103(a) as being unpatentable over U.S. Patent No. 5,861,882 to Sprenger et al. ("Sprenger"), over U.S. Patent No. 6,560,557 to Carnahan et al. ("Carnahan"), and also over U.S. Patent No. 5,818,446 to Bertram et al. (Bertram). Applicant respectfully traverses this rejection.

As per amended claim 43, the claim recites in part, "receiving user input specifying the data source, wherein the user input is received to a program development environment during creation of an executable program". The Office Action relies on Sprenger to teach this element of claim 43. However, Sprenger does not teach the creation of an executable program, i.e., a program that is executable by a processor.

Sprenger relates generally to testing test elements that are coupled to a bus of a computer system. Icons representing the test elements are displayed in an "Equipment Shelf" window. The user drags icons from this window onto a "Test Bench" window and draws lines between I/O ports on the icons to create a graphical description of a desired test circuit (Abstract; Col. 7, lines 45-50). This graphical description of the test circuit is not an executable program, i.e., is not executable by a processor. Instead, Sprenger describes that the graphical description is used to identify switch commands for configuring I/O ports of the test elements. The identified switch commands are transmitted on the bus to accomplish the graphically described test circuit. For example, see Col. 8, lines 40-49:

> The act of graphically selecting the icons and connecting their I/O port symbols provides information so that computer 12 can create a net list of the physical ports to be connected on the physical test elements. Computer 12 retrieves from memory the bus control signals which set switches 28 to couple the corresponding I/O ports of the corresponding test elements 26, DUT interface 30 and DUT 32 together in the manner determined by the graphical connections in circuit 54 on Test Bench window 42.

Also see Col. 13, lines 41-55:

> When the graphical connection tool selected in step 112 makes a successful graphical connection between I/O port symbols, then in step 128 the corresponding physical I/O connections to the selected test elements are identified and, from memory 18, the commands are identified which must be transmitted on bus 25 to switches 28 to accomplish the connection described in graphical format by the user in step 120. Thus, on completion of step 128, computer 12 knows which I/O ports on which test elements are desired to be coupled to which other I/O ports on which other test elements present on Test Bench window 42, and has retrieved the corresponding switch control commands from memory necessary to create physical connections via bus 25 corresponding to the graphical connections.

Thus, Sprenger nowhere describes that the graphical description of the test circuit is an executable program that is executable by a processor, and in fact, teaches away from the graphical description being an executable program, as described in the passages cited above. Instead, the graphical description is used to identify and retrieve switch commands to transmit on the bus to configure the test elements as described by the graphical description. It is noted that the graphical description is not used to generate the

switch commands. Instead, Sprenger teaches that the switch commands already exist in memory, and the graphical description simply identifies which switch commands to retrieve.

Applicant thus respectfully submits that Sprenger does not teach the element recited in amended claim 43 of, "receiving user input specifying the data source, wherein the user input is received to a program development environment during creation of an executable program".

Claim 43 has been amended to further recite the element of, "programmatically selecting a GUI element for inclusion in the program after receiving the user input". The Office Action relies on Bertram to teach this element of claim 43. However, Bertram does not teach programmatically selecting a GUI element for inclusion in a program.

Bertram relates generally to the presentation of a user interface on a computer. In response to a user preference or when data content changes, the user interface presented to the user is changed (Abstract; Col. 6, lines 20-38). Bertram teaches that, "In the invention, any user interface is changed by simply removing the currently active user interface and control code being executed in the processor and replacing it with a new user interface and control code without affecting the data being displayed" (Col. 7, lines 26-35). Bertram clearly teaches that "the entire user interface is changed" (Abstract). Applicant notes that selection of an entire user interface is not at all the same as selection of a GUI element. A GUI element is not a user interface, but is an element within a graphical user interface.

Furthermore, the particular GUI elements in each of Betram's user interfaces are apparently fixed. Bertram simply teaches that each user interface is registered, and at an appropriate time, is displayed. Bertram contains no teaching whatsoever regarding programmatically selecting a particular GUI element for inclusion in a user interface, and in particular, does not teach programmatically selecting a GUI element after receiving user input specifying a data source.

Applicant thus respectfully submits that Bertram does not teach, "programmatically selecting a GUI element for inclusion in the program after receiving the user input", as recited in claim 43.

Thus, the combination of the Sprenger, Carnahan, and Bertram references cited in the Office Action do not teach all of the elements of amended claim 43. Applicant thus submits that claim 43 is allowable, for at least the reasons discussed above.

Furthermore, Applicant submits that Sprenger, Carnahan, and Bertram are not properly combinable, and that even if the references were combined, they still would not produce the combination of elements recited in claim 43.

For example, as described above, the test elements in Sprenger are coupled to a bus of a computer system. Sprenger teaches that switch commands are transmitted on the bus to configure the test elements as described by a graphical description. Carnahan teaches test elements coupled to a remote server located over a network. The Office Action proposes modifying Sprenger so that the test elements are coupled to a remote server located over a network as taught in Carnahan. However, in this case, Sprenger's switch commands would need to be transmitted over the network to the test elements coupled to the server instead of transmitting the switch commands over a bus as taught in Sprenger. The Office Action has not demonstrated that this is known in the art and that there would be a reasonable expectation of success in combining the references in this manner. Applicant respectfully requests that the Examiner provide a reference that teaches transmitting switch commands over a network to configure test elements coupled to a remote server.

Furthermore, even if Sprenger and Carnahan were combined, the combination still would not teach the element recited in claim 43 of, "receiving user input specifying the data source, wherein the user input is received to a program development environment during creation of an executable program, and wherein said receiving user input specifying the data source comprises receiving user input specifying a uniform resource locator (URL) of the data source".

As described above, Sprenger teaches that icons representing the test elements are displayed in an "Equipment Shelf" window, and the user drags icons from this window onto a "Test Bench" window. The Office Action has taken the element of "receiving user input specifying the data source" to mean that the user drags a test element icon from the "Equipment Shelf" window onto the "Test Bench" window and acknowledges that

"Sprenger, however, does not explicitly disclose that the user input specifying comprises a URL of the data source".

Carnahan teaches the use of a browser to access an active server page, e.g., by entering a URL. The Office Action relies on Carnahan to teach a URL but is not at all clear as to what purpose the URL would serve in the proposed combination of Sprenger and Carnahan. For example, the Office Action is not clear as to what the URL would be received by, what exactly the URL would reference, or what would appear on the user's computer in response to specifying the URL. Applicant assumes that the Examiner is suggesting that the user would enter a URL of the remote server into a web browser, and in response, there would appear on the user's computer an "Equipment Shelf" window including icons representing the test elements connected to the remote server. The user would then drag icons from the "Equipment Shelf" window onto a "Test Bench" window to create a graphical description of a test circuit as taught in Sprenger.

However, Applicant notes that in this scenario, although the user caused the "Equipment Shelf" window to appear by specifying a URL, the act of dragging the icons from the "Equipment Shelf" window onto the "Test Bench" window does not comprise specifying a URL. As noted above, the Office Action has taken the element of "receiving user input specifying the data source" to mean that the user drags a test element icon from the "Equipment Shelf" window onto the "Test Bench" window. Thus, even in the proposed combination, receiving the user input specifying the data source would still not comprise receiving user input specifying a URL of the data source, as recited in claim 43.

Furthermore, Applicant submits that a URL received by a web browser as taught in Carnahan does not constitute user input "received to a program development environment during creation of an executable program" as recited in claim 43. Carnahan contains no teaching at all regarding entering a URL into a program development environment during creation of an executable program.

Applicant thus submits that amended claim 43, and claims dependent thereon, are allowable over Sprenger, Carnahan, and Bertram. Inasmuch as the other independent claims include elements similar to those discussed above with respect to claim 43, Applicant submits that the other independent claims, and claims respectively dependent thereon, are also allowable over Sprenger, Carnahan, and Bertram.

Applicant also asserts that numerous ones of the dependent claims recite further distinctions over the cited art. However, since the independent claims have been shown to be patentably distinct, a further discussion of the dependent claims is not necessary at this time.

Claim 59 was rejected under 35 U.S.C. 102(b) as being anticipated by U.S. Patent No. 5,818,446 to Bertram et al. ("Bertram"). Applicant respectfully traverses this rejection. Claim 59 has been amended to read as follows:

> 59. (Currently Amended) A system for configuring a graphical user interface (GUI) element to subscribe to a data source, the system comprising:
> a display device;
> a processor;
> a memory medium coupled to the processor, wherein the memory medium stores a first program;
> wherein the processor is operable to execute the first program to:
> receive user input specifying the data source during creation of an executable second program, wherein said receiving user input specifying the data source comprises receiving user input specifying a uniform resource locator (URL) of the data source;
> programmatically select a GUI element for inclusion in the second program after receiving the user input, wherein the GUI element is selected based on a data type of the data source;
> display the selected GUI element in the second program after said programmatically selecting; and
> programmatically configure the GUI element to receive and display data from the specified data source.

Applicant respectfully submits that Bertram does not teach all elements of amended claim 59. For example, as discussed above with respect to claim 43, Bertram does not teach the element of programmatically selecting a GUI element (not a user interface) for inclusion in the second program after receiving the user input specifying the data source.

Furthermore, Bertram does not teach the element of receiving user input specifying the data source during creation of an executable second program. Bertram instead teaches receiving user input to a program such as a web browser. A web browser is not used to create executable programs.

Applicant therefore respectfully submits that amended claim 59 is allowable over Bertram.

CONCLUSION

In light of the foregoing amendments and remarks, Applicant submits the application is now in condition for allowance, and an early notice to that effect is requested.

If any extensions of time (under 37 C.F.R. § 1.136) are necessary to prevent the above referenced application(s) from becoming abandoned, Applicant(s) hereby petition for such extensions. If any fees are due, the Commissioner is authorized to charge said fees to Meyertons, Hood, Kivlin, Kowert & Goetzel PC Deposit Account No. 50-1505/5150-50800/JCH.

Also enclosed herewith are the following items:

☒ Request for Continued Examination

☒ Return Receipt Postcard

Respectfully submitted,

Jeffrey C. Hood
Reg. No. 35,198
ATTORNEY FOR APPLICANT(S)

Meyertons, Hood, Kivlin, Kowert & Goetzel PC
P.O. Box 398
Austin, TX 78767-0398
Phone: (512) 853-8800
Date: 5/23/2005 JCH/JLB